|  |
| --- |
| **用户界面设计与分析**  **实验报告**  项目名称 SSD4实验三  专业班级 软件工程1803  学 号 8209180334  姓 名 刘文龙  **实验成绩：**  **批阅教师：**  2019年 11 月 30 日 |

**一、实验要求**

编程-完整的日期/时间控制面板描述这项任务是创建一个实现完整的日期/时间控制面板的界面（在标准Windows 98控制面板之后进行仿制）。如下所示，该界面应包含一个带有两个窗格的选项卡式对话框控件：一个用于设置日期和时间，另一个用于设置时区。 “日期和时间”选项卡将是此任务的新选项，并且应类似于此处显示的选项卡-除非您不需要在右侧绘制钟面。对于“时区”选项卡，应使用在上一个编程分配中创建的界面组件和代码。总体而言，此接口维护九个数据：•第二设置（0至59）•分钟设置（0至59）•小时设置（24小时制，0到23）•每月的一天•一年中的月份• 那一年•当前所选时区的索引•当前所选时区与GMT的分钟偏移量•当前所选时区的标题您构造的界面应包含一个选项卡式对话框容器以及“确定”，“取消”和“应用”按钮。 “选项卡式对话框”容器应具有两个选项卡-为该分配新建的“日期和时间”选项卡，以及从上一个分配获取的“时区”选项卡。 “日期和时间”选项卡应包含两个框架（一个代表日期，一个代表时间）和一个标签，用于显示当前选定的时区（但不显示时区与GMT的偏移量）。 “日期”框架应允许用户指定月，年和日。应该使用ComboBox控件（月份）和TextBox和UpDown控件（年份）以及用于显示和更改日期的Calendar控件来完成此操作。此处使用的Calendar控件是具有以下设置的“ Microsoft Calendar Control 10.0”：ShowDateSelectors =“ False”，ShowTitle =“ False”，ShowHorizo​​ntalGrid =“ True”，ShowVerticalGrid =“ True”，DayLength =“系统（中）” ，GridCellEffect =“ flat”，并且BackColor设置为“ white”。为了使用此特定的Calendar控件，您可能需要使用工具->自定义工具箱-> COM组件界面将其添加到工具箱中。“日期和时间”选项卡的“时间”框架应包含一个PictureBox控件（该控件最终将包括一个时钟图，但现在可能留为空白）和一组用于小时，分钟，秒的控件以及上午/下午设置。小时，分钟和第二个界面中的每个界面都应包含一个TextBox和一个NumericUpDown控件。 am / pm界面应由两个单选按钮组成的组（控制阵列）。界面的最后部分应包含“确定”，“取消”和“应用”按钮。当用户按下“确定”或“取消”按钮时，您应该生成一份有关当前设置的小报告（类似于以下所示），然后退出界面（通过“卸载”表单）。如果用户按下“应用”按钮，则应生成类似的报告，但不退出界面。请注意，该接口下的某些数据项以与内部存储不同的形式呈现给用户。例如，一天中的小时在内部以24小时格式（0到23小时）存储，但以12小时格式（1到12小时加am / pm）显示给用户。同样，从多个点显示或控制某些项目。例如，当前时区既显示在“时区”选项卡中（此处未显示），又显示在“日期和时间”选项卡中作为标签标题的一部分。同样，当前月份设置既显示在ComboBox中，又显示在日历控件中。通过使用正确构造的数据访问器（设置/获取）方法，可以极大地简化更新这些多种形式的显示并在控件中输入的值和基础数据值之间进行同步的过程。像往常一样，每个控件应通过调用相应基础数据的set方法来响应其属性值的更改（可能在转换后（例如从12小时到24小时）。然后，设置的方法可以负责更新显示其值的每个控件（在检查以确保需要更新控件之后）。

**二、应用程序简介**

用Swing写出的纯前端界面，功能包括：

（1）日历

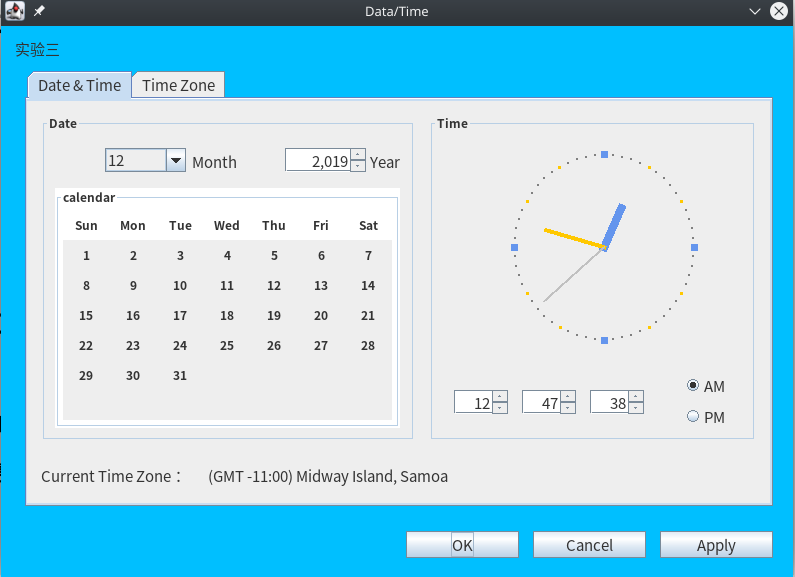
（2）钟表

（3）调整时间

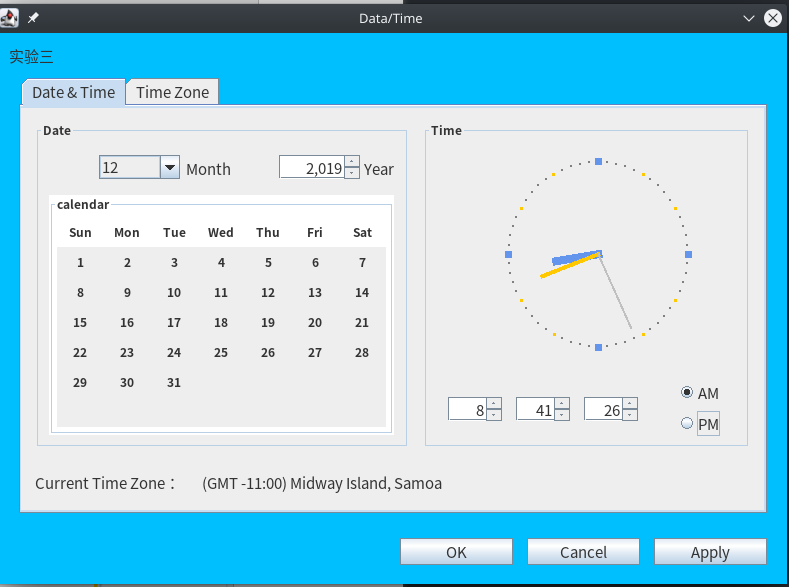
（4）选择时区

**三、应用程序页面展示及说明**

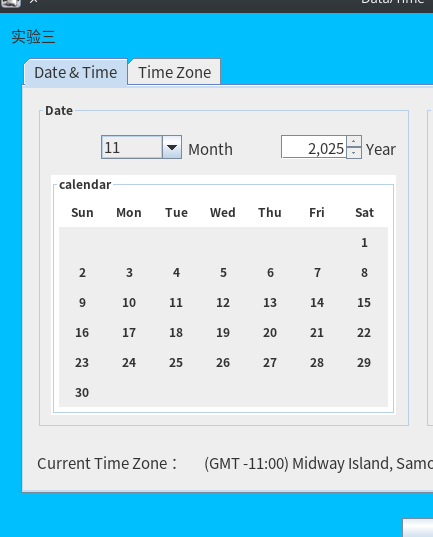
（1）首先是主页，主要包括两个卡片模块，一个是月日以及时间。，另一个是时区。与用户进行交互的也都在前端有相应的控件。



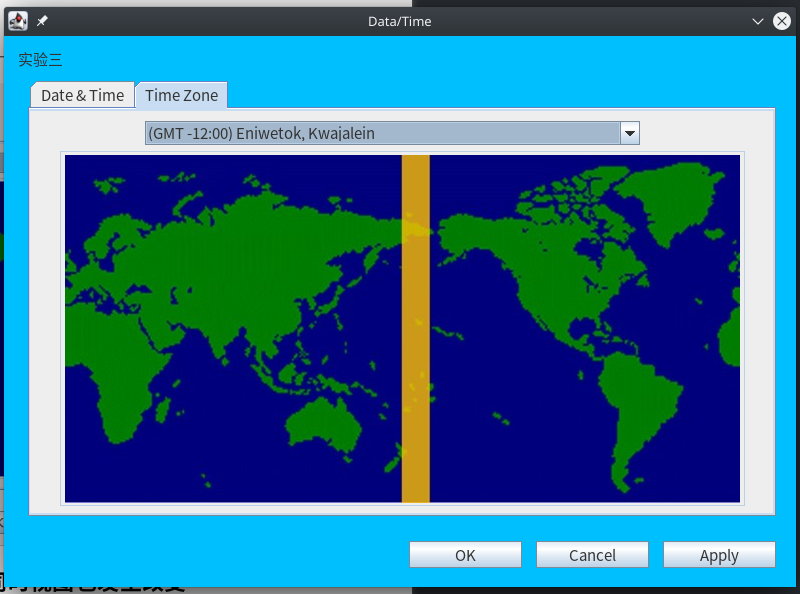
对时钟部分进行调整，然后相应的模块也会发生改变。



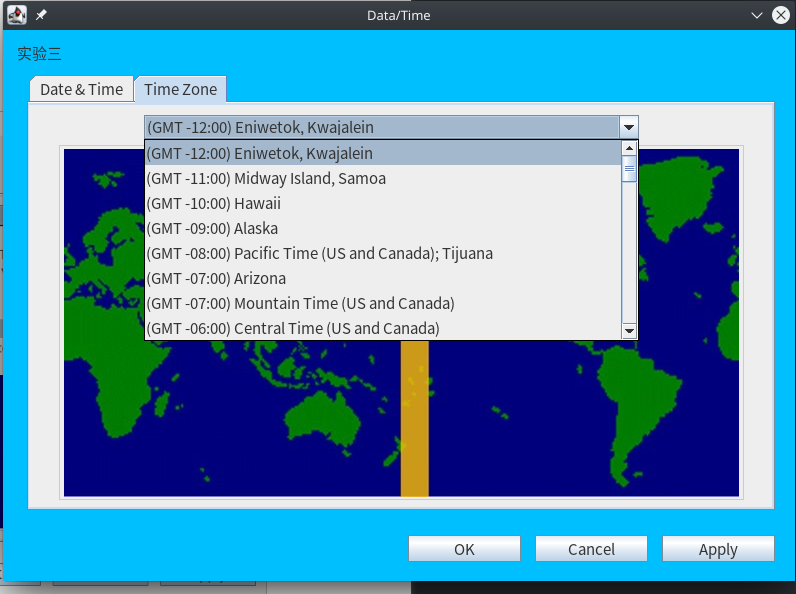
同样也可也实现月日的更改：



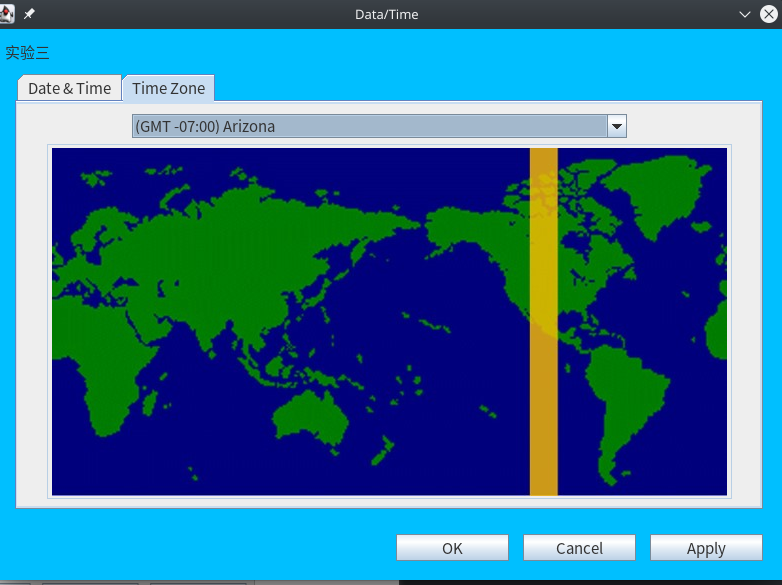
（2）关于时区的卡片



时区的交互组件：下拉列表



修改完成



**四、实验中遇到的问题**

1首先对于整体的程序构造理解了很久，用到了哪些组建，有什么用，怎么用。然后就是怎样进行布局

2然后是对于组件操作与反应的事件绑定。怎样让交互变得更加自然，更加贴合用户的使用习惯。让界面更加美观

3怎样将时间加进去，然后实现程序的运行，选择组件的包含。父子组件的使用。最后不断改进实现。

4有些地方遇到了一些意义不明的bug，最后人肉调试一波结束。

**五、实验心得**

本次实验让我明白了，一个优秀的前端，应该具备一下特点：美观，舒适，简单，引导，以及功能强大。然后就是对于敲代码的不熟练，应该加强对于java的熟悉程度。熟悉项目的开发流程，应该多在各大开源的网站进行学习，多练手，然后才能提高。最后就是要多学习，只有不断学习，才能够变强。

**六、项目源代码**

import java.awt.EventQueue;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.util.Calendar;

import java.util.Date;

import java.util.GregorianCalendar;

import java.util.TimeZone;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.Border;

import javax.swing.border.EmptyBorder;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import javax.swing.JTabbedPane;

import javax.swing.BorderFactory;

import javax.swing.ButtonGroup;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JSpinner;

import javax.swing.JRadioButton;

import javax.swing.JComboBox;

public class DateAndTime extends JFrame {

// 各面板

private JPanel contentPane;

private JTabbedPane tabbedPane;

private JPanel DateAndTimePane;

private JPanel DateAndTime\_Time;

private Border DateAndTime\_TimeBorder;

private JPanel DateAndTime\_Date;

private Border DateAndTime\_DateBorder;

private JPanel TimeZonePane;

// AM,PM单选框

static JRadioButton rdbtnAm;

static JRadioButton rdbtnPm;

// 是否更改并停止时间

private boolean flag;

// 数字电子时钟的三个显示栏

private JSpinner hourSpinner;

private JSpinner minuteSpinner;

private JSpinner secondSpinner;

// 获取小时，分钟，秒

private int hourOfDay, hour, minute, second;

// 今天的日期对象

static GregorianCalendar now;

// 时钟标签，上面画的是圆形时钟

private ClockLabel clockLabel;

// 获取年份，月份

private int year, month;

// 日历面板

private CalendarPanel calendarPanel;

// 三个按钮

private JButton btnOk, btnCanceled, btnApply;

// 时区选择

private JComboBox<String> timezoneBox;

// 时区图面板

private JPanel timezonepanel;

// 时区图

private JLabel timezone;

// 数字时钟线程

private Thread mathticClock;

// 时区信息数组

private String[][] tz;

private String[] GMT;

private JLabel timeZoneLabel;

// 日期

Date date;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

DateAndTime frame = new DateAndTime();

frame.setTitle("Data/Time");

frame.setFont(new Font("微软雅黑", Font.PLAIN, 15));

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public DateAndTime() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 792, 580);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

// 窗口不可调

setResizable(false);

// 卡片布局

tabbedPane = new JTabbedPane(JTabbedPane.TOP);

tabbedPane.setFont(new Font("微软雅黑", Font.PLAIN, 15));

tabbedPane.setBounds(24, 43, 748, 437);

contentPane.add(tabbedPane);

// 处理卡片布局的第一张卡片——————————DateAndTime

DateAndTimePane = new JPanel();

tabbedPane.addTab("Date & Time", null, DateAndTimePane, null);

DateAndTimePane.setLayout(null);

// 需要先获得当前时间

now = new GregorianCalendar();

date = Calendar.getInstance().getTime();

// 属于第一张卡片的右边面板TimePanel

DateAndTime\_Time = new JPanel();

DateAndTime\_Time.setBounds(402, 13, 327, 327);

DateAndTimePane.add(DateAndTime\_Time);

DateAndTime\_Time.setLayout(null);

// 设置JPanel的标题和边框

DateAndTime\_TimeBorder = BorderFactory.createTitledBorder("Time");

DateAndTime\_Time.setBorder(DateAndTime\_TimeBorder);

// 属于第一张卡片的左边面板DatePanel

DateAndTime\_Date = new JPanel();

DateAndTime\_Date.setBounds(14, 13, 374, 327);

DateAndTimePane.add(DateAndTime\_Date);

DateAndTime\_Date.setLayout(null);

// 设置JPanel的标题和边框

DateAndTime\_DateBorder = BorderFactory.createTitledBorder("Date");

DateAndTime\_Date.setBorder(DateAndTime\_DateBorder);

// ————————电子数字时钟的 时、分、秒位————————

flag = true;

hour = now.get(Calendar.HOUR\_OF\_DAY) - 12 > 0 ? (now.get(Calendar.HOUR\_OF\_DAY) - 12) : now.get(Calendar.HOUR);// 12小时制

minute = now.get(Calendar.MINUTE);

second = now.get(Calendar.SECOND);

changeListen change = new changeListen();

hourSpinner = new JSpinner();

hourSpinner.addChangeListener(change);

hourSpinner.setFont(new Font("微软雅黑", Font.PLAIN, 15));

hourSpinner.setBounds(25, 276, 54, 24);

DateAndTime\_Time.add(hourSpinner);

minuteSpinner = new JSpinner();

minuteSpinner.addChangeListener(change);

minuteSpinner.setFont(new Font("微软雅黑", Font.PLAIN, 15));

minuteSpinner.setBounds(93, 276, 54, 24);

DateAndTime\_Time.add(minuteSpinner);

secondSpinner = new JSpinner();

secondSpinner.addChangeListener(change);

secondSpinner.setFont(new Font("微软雅黑", Font.PLAIN, 15));

secondSpinner.setBounds(161, 276, 54, 24);

DateAndTime\_Time.add(secondSpinner);

// ————————电子数字时钟的 时、分、秒位————————

// 两个单选框AM/PM

rdbtnClick rdbtnclick = new rdbtnClick();

rdbtnAm = new JRadioButton("AM");

rdbtnAm.setFont(new Font("微软雅黑", Font.PLAIN, 15));

rdbtnAm.addActionListener(rdbtnclick);

rdbtnAm.setBounds(254, 258, 54, 27);

DateAndTime\_Time.add(rdbtnAm);

rdbtnPm = new JRadioButton("PM");

rdbtnPm.setFont(new Font("微软雅黑", Font.PLAIN, 15));

rdbtnPm.setBounds(254, 289, 54, 27);

DateAndTime\_Time.add(rdbtnPm);

// 让其实现单选

ButtonGroup buttonGroup = new ButtonGroup();

buttonGroup.add(rdbtnAm);

buttonGroup.add(rdbtnPm);

// 得到当前时间

getTime();

// 创建一个动画时钟

clockLabel = new ClockLabel(now);

clockLabel.setBounds(25, 13, 258, 220);

clockLabel.setFocusable(false);

DateAndTime\_Time.add(clockLabel);

// 启动时钟

// ————————————————————日历————————————————————

calendarPanel = new CalendarPanel(date);// 14,74,346,240

calendarPanel.setBounds(14, 24, 346, 290);

DateAndTime\_Date.add(calendarPanel);

// ————————————————————日历————————————————————

// 处理卡片布局的第二张卡片————————Time Zone

TimeZonePane = new JPanel();

tabbedPane.addTab("Time Zone", null, TimeZonePane, null);

TimeZonePane.setLayout(null);

// 时区

timezoneBox = new JComboBox<String>();

// 初始化时区

initTimtZone();

timezoneBox.setFont(new Font("微软雅黑", Font.PLAIN, 15));

timezoneBox.addActionListener(new timezone());

timezoneBox.setBounds(115, 10, 495, 24);

TimeZonePane.add(timezoneBox);

// 时区图片

ImageIcon image = new ImageIcon("image/GMT+12.jpg");// 图片大小：540，318 675 348

timezonepanel = new JPanel();

timezonepanel.setBounds(30, 40, 685, 355);

timezonepanel.setBorder(BorderFactory.createTitledBorder(""));

TimeZonePane.add(timezonepanel);

timezonepanel.setLayout(null);

timezone = new JLabel();

timezone.setFont(new Font("微软雅黑", Font.PLAIN, 15));

timezone.setIcon(image);

timezone.setBounds(5, 4, 675, 348);

timezonepanel.add(timezone);

// 三个按钮

btnOk = new JButton("OK");

btnOk.addActionListener(new btnOk());

btnOk.setFont(new Font("微软雅黑", Font.PLAIN, 15));

btnOk.setBounds(405, 505, 113, 27);

contentPane.add(btnOk);

btnCanceled = new JButton("Cancel");

btnCanceled.addActionListener(new btnCanceled());

btnCanceled.setFont(new Font("微软雅黑", Font.PLAIN, 15));

btnCanceled.setBounds(532, 505, 113, 27);

contentPane.add(btnCanceled);

btnApply = new JButton("Apply");

btnApply.addActionListener(new btnApply());

btnApply.setFont(new Font("微软雅黑", Font.PLAIN, 15));

btnApply.setBounds(659, 505, 113, 27);

contentPane.add(btnApply);

// 打开窗口时获取焦点

this.addWindowListener(new WindowAdapter() {

public void windowOpened(WindowEvent e) {

btnOk.requestFocus();

}

});

// 创建文字标识

createLabel();

}

// 初始化时区

private void initTimtZone() {

tz = new String[2][50];

tz[0][0] = "(GMT -12:00) Eniwetok, Kwajalein ";

tz[0][1] = "(GMT -11:00) Midway Island, Samoa ";

tz[0][2] = "(GMT -10:00) Hawaii ";

tz[0][3] = "(GMT -09:00) Alaska ";

tz[0][4] = "(GMT -08:00) Pacific Time (US and Canada); Tijuana ";

tz[0][5] = "(GMT -07:00) Arizona ";

tz[0][6] = "(GMT -07:00) Mountain Time (US and Canada) ";

tz[0][7] = "(GMT -06:00) Central Time (US and Canada) ";

tz[0][8] = "(GMT -06:00) Mexico City, Tegucigalpa ";

tz[0][9] = "(GMT -06:00) Saskatchewan ";

tz[0][10] = "(GMT -05:00) Bogota, Lima ";

tz[0][11] = "(GMT -05:00) Eastern Time (US and Canada) ";

tz[0][12] = "(GMT -05:00) Indiana (East) ";

tz[0][13] = "(GMT -04:00) Atlantic Time (Canada) ";

tz[0][14] = "(GMT -04:00) Caracas, La Paz ";

tz[0][15] = "(GMT -03:30) Newfoundland ";

tz[0][16] = "(GMT -03:00) Brasilia ";

tz[0][17] = "(GMT -03:00) Buenos Aires, Georgetown ";

tz[0][18] = "(GMT -02:00) Mid-Atlantic ";

tz[0][19] = "(GMT -01:00) Azores, Cape Verde Is. ";

tz[0][20] = "(GMT +00:00) Greenwich Mean Time; Dublin, Edinburgh, London, Lisbon ";

tz[0][21] = "(GMT +00:00) Monrovia, Casablanca ";

tz[0][22] = "(GMT +01:00) Berlin, Stockhold, Rome, Bern, Brussels, Vienna ";

tz[0][23] = "(GMT +01:00) Paris, Madrid, Amsterdam ";

tz[0][24] = "(GMT +01:00) Prage, Warsaw, Budapest ";

tz[0][25] = "(GMT +02:00) Athens, Helsinki, Istanbul ";

tz[0][26] = "(GMT +02:00) Cairo ";

tz[0][27] = "(GMT +02:00) Eastern Europe ";

tz[0][28] = "(GMT +02:00) Harare, Pretoria ";

tz[0][29] = "(GMT +02:00) Israel ";

tz[0][30] = "(GMT +03:00) Baghdad, Kuwait, Nairobi, Riyadh ";

tz[0][31] = "(GMT +03:00) Moscow, St. Petersburgh, Kazan, Volgograd ";

tz[0][32] = "(GMT +03:00) Tehran ";

tz[0][33] = "(GMT +04:00) Abu Dhabi, Muscat, Tbilisi ";

tz[0][34] = "(GMT +04:30) Kabul ";

tz[0][35] = "(GMT +05:00) Islamabad, Karachi, Ekaterinburg, Tashkent ";

tz[0][36] = "(GMT +05:30) Bombay, Calcutta, Madras, New Delhi, Colombo ";

tz[0][37] = "(GMT +06:00) Almaty, Dhaka ";

tz[0][38] = "(GMT +07:00) Bangkok, Jakarta, Hanoi ";

tz[0][39] = "(GMT +08:00) Beijing, Chongqing, Urumqi ";

tz[0][40] = "(GMT +08:00) Hong Kong, Perth, Singapore, Taipei ";

tz[0][41] = "(GMT +09:00) Tokyo, Osaka, Sapporo, Seoul, Yakutsk ";

tz[0][42] = "(GMT +09:30) Adelaide ";

tz[0][43] = "(GMT +09:30) Darwin ";

tz[0][44] = "(GMT +10:00) Brisbane, Melbourne, Sydney ";

tz[0][45] = "(GMT +10:00) Guam, Port Moresby, Vladivostok ";

tz[0][46] = "(GMT +10:00) Hobart ";

tz[0][47] = "(GMT +11:00) Magadan, Solomon Is., New Caledonia ";

tz[0][48] = "(GMT +12:00) Fiji, Kamchatka, Marshall Is. ";

tz[0][49] = "(GMT +12:00) Wellington, Auckland ";

tz[1][0] = "image/GMT+12.jpg";

tz[1][1] = "image/GMT-11.jpg";

tz[1][2] = "image/GMT-10.jpg";

tz[1][3] = "image/GMT-9.jpg";

tz[1][4] = "image/GMT-8.jpg";

tz[1][5] = "image/GMT-7.jpg";

tz[1][6] = "image/GMT-7.jpg";

tz[1][7] = "image/GMT-6.jpg";

tz[1][8] = "image/GMT-6.jpg";

tz[1][9] = "image/GMT-6.jpg";;

tz[1][10] = "image/GMT-5.jpg";

tz[1][11] = "image/GMT-5.jpg";

tz[1][12] = "image/GMT-5.jpg";

tz[1][13] = "image/GMT-4.jpg";

tz[1][14] = "image/GMT-4.jpg";

tz[1][15] = "image/GMT-4.jpg";;

tz[1][16] = "image/GMT-3.jpg";

tz[1][17] = "image/GMT-3.jpg";

tz[1][18] = "image/GMT-2.jpg";

tz[1][19] = "image/GMT-1.jpg";

tz[1][20] = "image/GMT.jpg";

tz[1][21] = "image/GMT.jpg";

tz[1][22] = "image/GMT+1.jpg";

tz[1][23] = "image/GMT+1.jpg";

tz[1][24] = "image/GMT+1.jpg";

tz[1][25] = "image/GMT+2.jpg";

tz[1][26] = "image/GMT+2.jpg";

tz[1][27] = "image/GMT+2.jpg";

tz[1][28] = "image/GMT+2.jpg";

tz[1][29] = "image/GMT+2.jpg";

tz[1][30] = "image/GMT+3.jpg";

tz[1][31] = "image/GMT+3.jpg";

tz[1][32] = "image/GMT+3.jpg";

tz[1][33] = "image/GMT+4.jpg";

tz[1][34] = "image/GMT+4.jpg";

tz[1][35] = "image/GMT+5.jpg";

tz[1][36] = "image/GMT+6.jpg";

tz[1][37] = "image/GMT+6.jpg";

tz[1][38] = "image/GMT+7.jpg";

tz[1][39] = "image/GMT+8.jpg";

tz[1][40] = "image/GMT+8.jpg";

tz[1][41] = "image/GMT+9.jpg";

tz[1][42] = "image/GMT+10.jpg";

tz[1][43] = "image/GMT+10.jpg";

tz[1][44] = "image/GMT+10.jpg";

tz[1][45] = "image/GMT+10.jpg";

tz[1][46] = "image/GMT+10.jpg";

tz[1][47] = "image/GMT+11.jpg";

tz[1][48] = "image/GMT+12.jpg";

tz[1][49] = "image/GMT+12.jpg";

GMT = new String[50];

GMT[0] = "GMT-12";

GMT[1] = "GMT-11";

GMT[2] = "GMT-10";

GMT[3] = "GMT-9";

GMT[4] = "GMT-8";

GMT[5] = "GMT-7";

GMT[6] = "GMT-7";

GMT[7] = "GMT-6";

GMT[8] = "GMT-6";

GMT[9] = "GMT-6";

GMT[10] = "GMT-5";

GMT[11] = "GMT-5";

GMT[12] = "GMT-5";

GMT[13] = "GMT-4";

GMT[14] = "GMT-4";

GMT[15] = "GMT-3";

GMT[16] = "GMT-3";

GMT[17] = "GMT-3";

GMT[18] = "GMT-2";

GMT[19] = "GMT-1";

GMT[20] = "GMT+0";

GMT[21] = "GMT+0";

GMT[22] = "GMT+0";

GMT[23] = "GMT+1";

GMT[24] = "GMT+1";

GMT[25] = "GMT+2";

GMT[26] = "GMT+2";

GMT[27] = "GMT+2";

GMT[28] = "GMT+2";

GMT[29] = "GMT+2";

GMT[30] = "GMT+3";

GMT[31] = "GMT+3";

GMT[32] = "GMT+3";

GMT[33] = "GMT+4";

GMT[34] = "GMT+5";

GMT[35] = "GMT+5";

GMT[36] = "GMT+6";

GMT[37] = "GMT+6";

GMT[38] = "GMT+7";

GMT[39] = "GMT+8";

GMT[40] = "GMT+8";

GMT[41] = "GMT+9";

GMT[42] = "GMT+10";

GMT[43] = "GMT+10";

GMT[44] = "GMT+10";

GMT[45] = "GMT+10";

GMT[46] = "GMT+10";

GMT[47] = "GMT+11";

GMT[48] = "GMT+12";

GMT[49] = "GMT+12";

for(int i = 0;i < tz[0].length;i++){

timezoneBox.addItem(tz[0][i]);

}

}

// 时区选择框的监事件

class timezone implements ActionListener {

@SuppressWarnings("deprecation")

@Override

public void actionPerformed(ActionEvent e) {

for(int i = 0;i < tz[0].length;i++){

if (timezoneBox.getSelectedItem().equals(tz[0][i])) {

TimeZone.setDefault(TimeZone.getTimeZone(GMT[i]));

date = Calendar.getInstance().getTime();

System.out.println(date);

timezone.setIcon(new ImageIcon(tz[1][i]));

timeZoneLabel.setText(tz[0][i]);

}

}

// 获取当前时区时间

year = date.getYear();

month = date.getMonth();

hourOfDay = date.getHours();

minute = date.getMinutes();

second = date.getSeconds();

hour = (hourOfDay - 12) > 0 ? (hourOfDay - 12) : hourOfDay;

System.out.println(hour + " " + hourOfDay);

// 复原now 对象

now.set(Calendar.YEAR, year);

now.set(Calendar.MONTH, month);

now.set(Calendar.HOUR, hour);

now.set(Calendar.MINUTE, minute);

now.set(Calendar.SECOND, second);

// 复原各控件内的值

calendarPanel.initMonthAndYear();

hourSpinner.setValue(hour);

minuteSpinner.setValue(minute);

secondSpinner.setValue(second);

if (hourOfDay > 12) {

rdbtnPm.setSelected(true);

} else {

rdbtnAm.setSelected(true);

}

}

}

// AM,PM点击事件

class rdbtnClick implements ActionListener {

@SuppressWarnings("deprecation")

@Override

public void actionPerformed(ActionEvent e) {

// 确定更改时间

ClockLabel.flag = false;

ClockLabel.clockThread.suspend();

mathticClock.suspend();

}

}

// OK按钮事件

class btnOk implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

}

// cancel按钮事件

class btnCanceled implements ActionListener {

@SuppressWarnings("deprecation")

@Override

public void actionPerformed(ActionEvent e) {

// 恢复到GMT+8 的北京时间

TimeZone.setDefault(TimeZone.getTimeZone("GMT+8"));

date = now.getTime();

// 恢复时区

timezoneBox.setSelectedItem(tz[0][1]);

timezone.setIcon(new ImageIcon(tz[1][0]));

// 获取当前系统时间

year = Calendar.getInstance().get(Calendar.YEAR);

month = Calendar.getInstance().get(Calendar.MONTH);

hourOfDay = Calendar.getInstance().get(Calendar.HOUR\_OF\_DAY);

minute = Calendar.getInstance().get(Calendar.MINUTE);

second = Calendar.getInstance().get(Calendar.SECOND);

hour = (hourOfDay - 12) > 0 ? (hourOfDay - 12) : hourOfDay;

// 复原now 对象

now.set(Calendar.YEAR, year);

now.set(Calendar.MONTH, month);

now.set(Calendar.HOUR, hour);

now.set(Calendar.MINUTE, minute);

now.set(Calendar.SECOND, second);

// 复原各控件内的值

calendarPanel.initMonthAndYear();

hourSpinner.setValue(hour);

minuteSpinner.setValue(minute);

secondSpinner.setValue(second);

if (hourOfDay > 12) {

rdbtnPm.setSelected(true);

} else {

rdbtnAm.setSelected(true);

}

// 恢复线程

ClockLabel.clockThread.resume();

mathticClock.resume();

}

}

// apply按钮事件

class btnApply implements ActionListener {

@SuppressWarnings("deprecation")

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

ClockLabel.flag = false;

ClockLabel.clockThread.resume();

mathticClock.resume();

}

}

// 电子数字时钟的监听事件

class changeListen implements ChangeListener {

@Override

public void stateChanged(ChangeEvent e) {

if ((Integer) hourSpinner.getValue() >= 13) {

hourSpinner.setValue(1);

}

if ((Integer) hourSpinner.getValue() <= 0) {

hourSpinner.setValue(12);

}

// 设置系统时间

now.set(Calendar.HOUR, (int) hourSpinner.getValue());

if ((Integer) minuteSpinner.getValue() >= 60) {

minuteSpinner.setValue(0);

hourSpinner.setValue((Integer) (hourSpinner.getValue()) + 1);

}

if ((Integer) minuteSpinner.getValue() < 0) {

minuteSpinner.setValue(59);

hourSpinner.setValue((Integer) hourSpinner.getValue() - 1);

}

// 设置系统时间

now.set(Calendar.MINUTE, (int) minuteSpinner.getValue());

if ((Integer) secondSpinner.getValue() >= 60) {

secondSpinner.setValue(0);

minuteSpinner.setValue((Integer) (minuteSpinner.getValue()) + 1);

}

if ((Integer) secondSpinner.getValue() < 0) {

secondSpinner.setValue(59);

minuteSpinner.setValue((Integer) minuteSpinner.getValue() - 1);

}

// 设置系统时间

now.set(Calendar.SECOND, (int) secondSpinner.getValue());

}

}

// 文字标识

void createLabel() {

JLabel lblPonzi = new JLabel("SSD4");

lblPonzi.setFont(new Font("微软雅黑", Font.PLAIN, 15));

lblPonzi.setBounds(14, 13, 72, 18);

contentPane.add(lblPonzi);

JLabel lblExample = new JLabel("Example (Write By LGX)");

lblExample.setFont(new Font("微软雅黑", Font.PLAIN, 15));

lblExample.setBounds(100, 13, 500, 18);

contentPane.add(lblExample);

JLabel lblNewLabel = new JLabel("Current Time Zone\uFF1A");

lblNewLabel.setFont(new Font("微软雅黑", Font.PLAIN, 15));

lblNewLabel.setBounds(14, 358, 189, 32);

DateAndTimePane.add(lblNewLabel);

JLabel label = new JLabel("Year");

label.setBounds(315, 13, 72, 20);

calendarPanel.add(label);

label.setFont(new Font("微软雅黑", Font.PLAIN, 15));

JLabel label\_1 = new JLabel("Month");

label\_1.setBounds(137, 13, 72, 20);

calendarPanel.add(label\_1);

label\_1.setFont(new Font("微软雅黑", Font.PLAIN, 15));

timeZoneLabel = new JLabel(tz[0][1]);

timeZoneLabel.setFont(new Font("微软雅黑", Font.PLAIN, 15));

timeZoneLabel.setBounds(181, 359, 400, 31);

DateAndTimePane.add(timeZoneLabel);

}

// 获得当前时间

@SuppressWarnings("deprecation")

void getTime() {

// 设置时间

hourSpinner.setValue((date.getHours() - 12) > 0 ? (date.getHours() - 12) : date.getHours());

minuteSpinner.setValue(date.getMinutes());

secondSpinner.setValue(date.getSeconds());

// 设置当前时间是 AM 还是 PM

if (now.get(Calendar.HOUR\_OF\_DAY) >= 12) {

rdbtnPm.setSelected(true);

} else {

rdbtnAm.setSelected(true);

}

// 创建一个新线程！

mathticClock = new MathticClock();

mathticClock.start();

}

class MathticClock extends Thread {

@Override

public void run() {

while (flag) {

secondSpinner.setValue((Integer) (secondSpinner.getValue()) + 1);

try {

Thread.sleep(1000);// 休眠1秒

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

**CalendarBean类**

package control;

import java.util.Calendar;

public class CalendarBean {

int year, month;

public void setYear(int year) {

this.year = year;

}

public int getYear() {

return year;

}

public void setMonth(int month) {

this.month = month;

}

public int getMonth() {

return month;

}

public String[] getCalendar() {

String a[] = new String[42];

Calendar date = Calendar.getInstance();

date.set(year, month - 1, 1);

int week = date.get(Calendar.DAY\_OF\_WEEK) - 1;

int day = 0;

// 判断大月份

if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12) {

day = 31;

}

// 判断小月

if (month == 4 || month == 6 || month == 9 || month == 11) {

day = 30;

}

// 判断平年与闰年

if (month == 2) {

if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)) {

day = 29;

}

else {

day = 28;

}

}

// 赋值

for (int i = week, n = 1; i < week + day; i++) {

a[i] = String.valueOf(n);

n++;

}

return a;

}

}

package ui;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.geom.Line2D;

import java.util.Calendar;

import java.util.GregorianCalendar;

import javax.swing.JLabel;

import javax.swing.JRadioButton;

//自定义时钟标签，画一个圆形的时钟

public class ClockLabel extends JLabel {

// 时钟标签的宽度和高度

private final int WIDTH = 300;

private final int HEIGHT = 88 \* 3;

// 圆形时钟的X半径和Y半径

private final int CIRCLE\_X\_RADIUS = 90;

private final int CIRCLE\_Y\_RADIUS = 93;

// 圆形时钟的原点

private final int CIRCLE\_X = 150;

private final int CIRCLE\_Y = 120;

// 圆形时钟指针的长度

private final int HOUR\_LENGTH = 14 \* 3;

private final int MIN\_LENGTH = 60;

private final int SEC\_LENGTH = 27 \* 3;

// 当前时针所处的角度

double arcHour = 0;

// 当前分针所处的角度

double arcMin = 0;

// 当前秒针所处的角度

double arcSec = 0;

// 时钟线程

static Thread clockThread = null;

// 是否更改了时间

static boolean flag;

public ClockLabel(GregorianCalendar now) {

// 设置时钟标签的大小

this.setSize(WIDTH, HEIGHT);

// 获取时针、分针、秒针当前的角度

arcHour = now.get(Calendar.HOUR) \* (360.0 / 12) + now.get(Calendar.MINUTE) \* (360.0 / 12 / 60)

+ now.get(Calendar.SECOND) \* (360.0 / 12 / 60 / 60);

arcMin = now.get(Calendar.MINUTE) \* (360.0 / 60) + now.get(Calendar.SECOND) \* (360.0 / 60 / 60);

arcSec = now.get(Calendar.SECOND) \* (360.0 / 60);

flag = true;

clockThread = new ClockThread();

clockThread.start();

}

public void paint(Graphics g1) {

// Graphics2D继承Graphics，比Graphics提供更丰富的方法

Graphics2D g = (Graphics2D) g1;

/\*\* \*\*\*画圆形时钟的刻度，每6度便有一个刻度\*\*\*\* \*/

for (int i = 0; i < 360; i = i + 6) {

g.setColor(Color.gray);

// 设置画笔的宽度为2

g.setStroke(new BasicStroke(2));

// 画刻度

if (i % 90 == 0) {

// 对于0，3，6，9点位置，使用一个大的蓝色刻度

g.setColor(new Color(100, 149, 237));

g.setStroke(new BasicStroke(7));// 画笔宽度为5

// 当起点和终点一样时，画的就是点

g.drawLine(CIRCLE\_X + (int) (Math.cos(i \* Math.PI / 180) \* CIRCLE\_X\_RADIUS),

CIRCLE\_Y + (int) (Math.sin(i \* Math.PI / 180) \* CIRCLE\_Y\_RADIUS),

CIRCLE\_X + (int) (Math.cos(i \* Math.PI / 180) \* CIRCLE\_X\_RADIUS),

CIRCLE\_Y + (int) (Math.sin(i \* Math.PI / 180) \* CIRCLE\_Y\_RADIUS));

} else if (i % 30 == 0) {

// 如果角度处于小时的位置，而且还不在0，3，6，9点时，画红色的小刻度

g.setColor(Color.orange);

g.setStroke(new BasicStroke(3));// 画笔宽度为3

g.drawLine(CIRCLE\_X + (int) (Math.cos(i \* Math.PI / 180) \* CIRCLE\_X\_RADIUS),

CIRCLE\_Y + (int) (Math.sin(i \* Math.PI / 180) \* CIRCLE\_Y\_RADIUS),

CIRCLE\_X + (int) (Math.cos(i \* Math.PI / 180) \* CIRCLE\_X\_RADIUS),

CIRCLE\_Y + (int) (Math.sin(i \* Math.PI / 180) \* CIRCLE\_Y\_RADIUS));

} else {

// 其他位置就画小刻度

g.setColor(Color.gray);

g.drawLine(CIRCLE\_X + (int) (Math.cos(i \* Math.PI / 180) \* CIRCLE\_X\_RADIUS),

CIRCLE\_Y + (int) (Math.sin(i \* Math.PI / 180) \* CIRCLE\_Y\_RADIUS),

CIRCLE\_X + (int) (Math.cos(i \* Math.PI / 180) \* CIRCLE\_X\_RADIUS),

CIRCLE\_Y + (int) (Math.sin(i \* Math.PI / 180) \* CIRCLE\_Y\_RADIUS));

}

}

/\*\* \*\*\*\*\*\* 画时钟的指针 \*\*\*\*\*\*\*\* \*/

// 画时针

Line2D.Double lh = new Line2D.Double(CIRCLE\_X, CIRCLE\_Y,

CIRCLE\_X + Math.cos((arcHour - 90) \* Math.PI / 180) \* HOUR\_LENGTH,

CIRCLE\_Y + Math.sin((arcHour - 90) \* Math.PI / 180) \* HOUR\_LENGTH);

// 设置画笔宽度和颜色

g.setStroke(new BasicStroke(8));

g.setColor(new Color(100, 149, 237));

// 利用Graphics2D的draw方法画线

g.draw(lh);

// 画分针

Line2D.Double lm = new Line2D.Double(CIRCLE\_X, CIRCLE\_Y,

CIRCLE\_X + Math.cos((arcMin - 90) \* Math.PI / 180) \* MIN\_LENGTH,

CIRCLE\_Y + Math.sin((arcMin - 90) \* Math.PI / 180) \* MIN\_LENGTH);

g.setStroke(new BasicStroke(4));

g.setColor(Color.orange);

g.draw(lm);

// 画秒针

Line2D.Double ls = new Line2D.Double(CIRCLE\_X, CIRCLE\_Y,

CIRCLE\_X + Math.cos((arcSec - 90) \* Math.PI / 180) \* SEC\_LENGTH,

CIRCLE\_Y + Math.sin((arcSec - 90) \* Math.PI / 180) \* SEC\_LENGTH);

g.setStroke(new BasicStroke(2));

g.setColor(Color.lightGray);

g.draw(ls);

}

class ClockThread extends Thread{

@Override

public void run() {

while (true) {

try {

// 获取时针、分针、秒针当前的角度

arcHour = DateAndTime.now.get(Calendar.HOUR) \* (360.0 / 12) + DateAndTime.now.get(Calendar.MINUTE) \* (360.0 / 12 / 60)

+ DateAndTime.now.get(Calendar.SECOND) \* (360.0 / 12 / 60 / 60);

arcMin = DateAndTime.now.get(Calendar.MINUTE) \* (360.0 / 60) +DateAndTime. now.get(Calendar.SECOND) \* (360.0 / 60 / 60);

arcSec = DateAndTime.now.get(Calendar.SECOND) \* (360.0 / 60);

// 设置AM，PM的选择

APset(DateAndTime.rdbtnAm, DateAndTime.rdbtnPm);

// 重画时钟标签

repaint();

// 等待0.1秒钟

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public void APset(JRadioButton rdbtnAm, JRadioButton rdbtnPm){

// 设置AM，PM的选择

if (flag) {

if (Calendar.getInstance().get(Calendar.HOUR\_OF\_DAY) > 12) {

rdbtnPm.setSelected(true);

} else {

rdbtnAm.setSelected(true);

}

}

}

}

package ui;

import java.awt.\*;

import java.awt.event.\*;

import java.util.Date;

import java.util.Calendar;

import javax.swing.\*;

import javax.swing.border.Border;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import control.CalendarBean;

public class CalendarPanel extends JPanel {

JLabel labelDay[] = new JLabel[42];

JLabel titleName[] = new JLabel[7];

JButton button = new JButton();

String name[] = { " Sun ", " Mon ", " Tue ", " Wed ", " Thu ", " Fri ", " Sat " };

int year, month, dayOfMonth; // 启动程序显示的日期信息

CalendarBean calendar;

// 月份选择框

JComboBox<String> MonthBox;

// 年份显示栏

JSpinner yearSpinner;

public CalendarPanel(Date date) {

this.setLayout(null);

mouseClick mouse = new mouseClick();

calendar = new CalendarBean();

calendar.setYear(date.getYear());

calendar.setMonth(date.getMonth());

String day[] = calendar.getCalendar();

// 获得当前日期

dayOfMonth = date.getDate();

JPanel pCenter = new JPanel();

pCenter.setBounds(0, 50, 345, 240);

pCenter.setBackground(Color.white);

Border border = BorderFactory.createTitledBorder("calendar");

pCenter.setBorder(border);

this.add(pCenter);// 窗口添加在中心区域

// 将pCenter的布局设置为7行7列的GridLayout 布局。

pCenter.setLayout(new GridLayout(7, 7));

// pCenter添加组件titleName[i]

for (int i = 0; i < 7; i++) {

titleName[i] = new JLabel(name[i], JLabel.CENTER);

pCenter.add(titleName[i]);

}

Rectangle rectangle = new Rectangle(20,20);

// pCenter添加组件labelDay[i]

for (int i = 0; i < 42; i++) {

labelDay[i] = new JLabel("", JLabel.CENTER);

labelDay[i].setBounds(rectangle);

// 使JLabel可着色

labelDay[i].setOpaque(true);

labelDay[i].addMouseListener(mouse);

pCenter.add(labelDay[i]);

}

int j = 0;

for (int i = 0; i < 42; i++) {

labelDay[i].setText(day[i]);

}

MonthBox = new JComboBox<String>();

MonthBox.setFont(new Font("微软雅黑", Font.PLAIN, 15));

MonthBox.setBounds(50, 10, 81, 24);

MonthBox.addActionListener(new MonthBox());

initMonthBox();

this.add(MonthBox, BorderLayout.NORTH);

yearSpinner = new JSpinner();

yearSpinner.setFont(new Font("微软雅黑", Font.PLAIN, 15));

yearSpinner.setBounds(230, 10, 81, 24);

yearSpinner.addChangeListener(new YearSpinner());

this.add(yearSpinner, BorderLayout.NORTH);

// 初始化年月

initMonthAndYear();

}

private void initMonthBox() {

MonthBox.addItem("1");

MonthBox.addItem("2");

MonthBox.addItem("3");

MonthBox.addItem("4");

MonthBox.addItem("5");

MonthBox.addItem("6");

MonthBox.addItem("7");

MonthBox.addItem("8");

MonthBox.addItem("9");

MonthBox.addItem("10");

MonthBox.addItem("11");

MonthBox.addItem("12");

}

// 监听月份栏

class MonthBox implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

month = Integer.parseInt(MonthBox.getSelectedItem().toString());

calendar.setMonth(month);

String day[] = calendar.getCalendar();

for (int i = 0; i < 42; i++) {

labelDay[i].setText(day[i]);

}

}

}

// 监听年份栏

class YearSpinner implements ChangeListener {

@Override

public void stateChanged(ChangeEvent e) {

// TODO Auto-generated method stub

calendar.setYear(Integer.parseInt(yearSpinner.getValue().toString()));

String day[] = calendar.getCalendar();

for (int i = 0; i < 42; i++) {

labelDay[i].setText(day[i]);

}

}

}

void initMonthAndYear() {

yearSpinner.setValue(Calendar.getInstance().get(Calendar.YEAR));

month = Calendar.getInstance().get(Calendar.MONTH);

switch (month) {

case Calendar.JANUARY:

MonthBox.setSelectedItem("1");

break;

case Calendar.FEBRUARY:

MonthBox.setSelectedItem("2");

break;

case Calendar.MARCH:

MonthBox.setSelectedItem("3");

break;

case Calendar.APRIL:

MonthBox.setSelectedItem("4");

break;

case Calendar.MAY:

MonthBox.setSelectedItem("5");

break;

case Calendar.JUNE:

MonthBox.setSelectedItem("6");

break;

case Calendar.JULY:

MonthBox.setSelectedItem("7");

break;

case Calendar.AUGUST:

MonthBox.setSelectedItem("8");

break;

case Calendar.SEPTEMBER:

MonthBox.setSelectedItem("9");

break;

case Calendar.OCTOBER:

MonthBox.setSelectedItem("10");

break;

case Calendar.NOVEMBER:

MonthBox.setSelectedItem("11");

break;

case Calendar.DECEMBER:

MonthBox.setSelectedItem("12");

break;

}

}

// 对鼠标事件的监听

class mouseClick implements MouseListener {

@Override

public void mouseClicked(MouseEvent e) {

for(int i = 0;i < 42;i++){

labelDay[i].setBackground(new Color(239,239,239));

if(e.getComponent().equals(labelDay[i]) && labelDay[i].getText() != null){

labelDay[i].setBackground(Color.GRAY);

}

}

}

@Override

public void mousePressed(MouseEvent e) {

}

@Override

public void mouseReleased(MouseEvent e) {

}

@Override

public void mouseEntered(MouseEvent e) {

}

@Override

public void mouseExited(MouseEvent e) {

}

}

}